Coursework Specification

CSC2021 - Software Engineering

TDD 1: Implementation of an appointment manager

Due 15:00:00, Friday, 29 Nov 2019 ~~(in 5 days)~~

Maximum mark is 30

CSC2021, Coursework 2: TDD Part I – Design and Development using TDD.

Introduction:

In this coursework you will use UML and then also some Test-Driven Development (TDD) to design and implement a simple student record management system for a University department. This Java class shall be referred to in this specification as the Record Manager.

The Record Manager is used to store the details of all of the department's students and their academic records. Academic records consist of a number of modules.

This coursework is worth 15% of the module mark for CSC2021.

We have a framework which you must use to create your test classes and provide a log of the tests you undertake.

Aim:

- To model a system using a UML class diagram
- To use TDD to implement some parts of a Java program

Learning/Skill Outcomes:

This coursework will enable you to practice the following skills:

- UML class designs
- TDD
- Java programming
- Brownfield coding (adding to an existing package structure)

Specification:

The coursework consists of 3 tasks. In the first, you will produce a UML class diagram for the RecordManager class. In Task 2, you will produce and test a first implementation of the Record Manager class using the provided Eclipse framework. In Task 3, you will use Test Driven Development (TDD) to implement advanced features of the RecordManager. In Tasks 2 and 3, you should implement the design you produced in Task 1 and **MUST** explain (in the header comments of your production code) if you have made changes.

Task 1:

Produce a UML class diagram for the system you intend to implement. You should call the class which has the following features RecordManager.java:

1. It must be possible to create multiple Record Managers (e.g. for different departments).

2. The data in the Record Manager only needs to exist during runtime - There does not need to be any persistence beyond the lifetime of the program.

3. The following information must be stored about each student by the Record Manager: unchangeable ID, name, postal address (as a single String), e-mail address, degree course title and the date they first enrolled on the course (as either a String or a Calendar) plus a list of modules. Each module consists of a module code (which you can assume is like Newcastle codes – three letters + four numbers), the module title, the module mark (as an integer between 1 and 100 – you may assume a student scores at least 1%) and the number of credits for that module. There is no limit on the number of modules a student can take.

4. A full list of all students and their details can be retrieved from a Record Manager.

5. A user can search the Record Manager (for completely matching) student names.
   - This should return a list of all matches and their details.

6. The Record Manager can be used to change a student's name and postal address.

7. The Record Manager can be used to add the details of a new module for a student, given that student's ID.

8. The Record Manager can be used to calculate a student's average mark as a double, given that student's ID. This average is simply the average of all the module marks currently held on a student's record.

Important note for Task 1:

You may use whatever drawing/design tool you wish to produce your UML diagram.

Task 2:

Implement **the features up to and including Feature 5 WITHOUT using TDD.** Test your code in JUnit classes whose name(s) start "Task2Test". **Failure to use JUnit or to name your classes correctly will lead to a mark of 0 for Task 2.**

You must use the Eclipse framework. Follow the instructions at the end of the assignment in order to obtain the framework and get it operational in Eclipse.

Using the framework

The framework contains two Java classes, "AbstractLoggingJUnitTest.java" and "ExampleLoggingTest.java". "AbstractLoggingJUnitTest.java" is a base class for your Task 3 tests and **must not be modified.** "ExampleLoggingTest.java" is an

example of a test class, as you may have gathered. You are advised to leave this unchanged but to use it as a template for your own test class/classes in Task 3.

All of your test classes must be in the folder src->com->example->tddCoursework as must your production code. You may create subfolders in this location, e.g. for production and test code.

Important notes for Task 2:

1. This is the first version of the code, so you should **not** use TDD to develop code for task 2: simply write the code as normal.
2. You **must** still test your code with a JUnit test class! The markers will look for at least one test class for these features. You should make sure your class signatures **do not extend** "AbstractLoggingJUnitTest" because you aren't logging the red-green cycle in this task.
3. Your Task 2 test class or classes must have Task2Test in their name and be saved in the folder shown above (src->com->example->tddCoursework). The markers will **only look in this folder** and will **only mark classes with Task2Test in their name** for Task 2.
4. You are advised to use some version control system to keep copies of the last working version of your code. That way, if you make a mistake, it is easy to check out the last working version again. This is not compulsory but is recommended and also applies to Task 3.

Task 3:

Implement **features 6, 7, and 8 plus a test for the following error case using JUnit and TDD.**

For the error case: A student cannot register for a module more than once. Test that your production code successfully prevents adding a module in Feature 7 whose code is identical to that of a module already taken by the student. If the module already exists on the student's record, the second occurrence should not be added to a student's list of modules. Note that the first occurrence should not be deleted from the student's record. It is up to you whether your production code should fail quietly in the error case or throw an IllegalArgumentException. Your production code should not print out an error message to the screen.

In the framework, you must ensure that **the class definition includes** "extends AbstractLoggingJUnitTest".

All of your test classes must be in the folder src->com->example->tddCoursework (or a subfolder of this) once again though this time the test class names are up to you.

Important notes for Task 3:

1. You **MUST follow TDD in Task 3!** The markers will use the test log HTML files provided by the framework to determine this so do **not** modify or delete them. You should be alternating between **RED phase** and **GREEN phase**. These transitions are visible in the log file (failed tests generate WARN log events, which are coloured red).
2. You **MUST** use a different test class to that used in Task 2.

3. The markers will **only look in src->com->example->tddCoursework for Task 3 test code and in the log directory for the log files.** Any test classes placed in other folders will be ignored.
4. We appreciate that mistakes happen in the development process. You will not be marked down if it takes several test runs for a test to move from the red to green phase due to bugs in your code – the important thing is that you follow the TDD process.

Marking Scheme:

This coursework is marked out of 30 with a breakdown as follows:

Task 1:

7 marks for your UML class diagram.

Task 2:

6 marks for evidence of testing the code produced in Task 2.

Task 3:

4 marks for use of the red-green cycle,

5 marks for writing/running appropriate tests,

2 marks for quality of test class and method names,

4 marks for structure and presentation of test methods,

2 marks for implementing the production code based on your design from Task 1 (this also covers code developed in Task 2).

Submission and Deadlines:

Your coursework must be submitted to the NESS system by the deadline specified. Note that NESS imposes deadlines rigorously, and even work that is a few seconds late (e.g. because of network delays caused by students all submitting at the last moment) will be flagged as late.

Coursework 2 (TDD part II) **has the same deadline but should be submitted separately.**

You must submit **two files**, a word document or pdf containing your UML diagram and a zip archive (.zip file) containing the src directory (this houses all of your production code and all of your test classes) and the log directory which contains the test logs. The easiest way to do this is to zip up the project directory, i.e. the "ncl_csc2021_tddCoursework-master" folder. The framework project directory that you initially download has all the components required for a valid submission (though no code that will achieve marks). NESS will require **both files** in order for you to submit. **It is YOUR responsibility to submit the correct files and to make sure the zip file contains the correct classes and log files.**

Obtaining/testing the framework:

Download and save the framework as a zip file from Blackboard, where it is in the "Assessment Information" section. Extract the zip file. This should give a folder

called “ncl_csc2021_tddCoursework-master”.

Import this project into Eclipse as follows:
1. Go to File->Import-> Existing Projects into Workspace.
2. Click “Next”.
3. Select root directory to location of downloaded project.
4. Click “Finish”.

Now test that it is working. You should do this **sooner rather than later** and ideally in a practical class so you can talk to the demonstrators if something goes wrong.

To test:

1. Open “ExampleLoggingTest.java”
2. Go to Run->Run As->JUnit Test (or use keyboard shortcut: F11)
3. Console output should be:

```
INFO : TestSuite [com.example.tddCoursework.ExampleLoggingTest] started
WARN :   Test [test]   failed with exception [Not yet implemented]
INFO :   Test [test2]   succeeded
INFO : TestSuite [com.example.tddCoursework.ExampleLoggingTest] finished
```

4. Additionally, there should be an html log file at “ncl_csc2021_tddCoursework-master/log”. Open the log file in your browser to check it has log data in it - it should be showing a log session that started very recently.
5. If you have reached this stage then it has worked.

Acknowledgements: Thanks to Dr John Shearer for providing the framework used for Tasks 2 and 3.

Last updated 15 Oct 2019 11:22:32

Submit this exercise